# 1. Install softwares at master

**root@master:~# vim all\_kube\_nodes.sh**

#!/bin/bash

#At master and worker both nodes

apt update

apt upgrade

echo "192.168.50.201 worker1.mahidul.com worker1" >> /etc/hosts

echo "192.168.50.200 master.mahidul.com master" >> /etc/hosts

#change the hostname as per nodes

#hostnamectl set-hostname master

#hostnamectl set-hostname worker1

modprobe br\_netfilter

modprobe overlay

cat << EOF | tee /etc/modules-load.d/k8s-modules.conf

br\_netfilter

overlay

EOF

cat << EOF | tee /etc/sysctl.d/k8s.conf

net.bridge.bridge-nf-call-ip6tables = 1

net.bridge.bridge-nf-call-iptables = 1

net.ipv4.ip\_forward = 1

EOF

sysctl --system

apt-get update ; apt-get install -y containerd

mkdir -p /etc/containerd

containerd config default | tee /etc/containerd/config.toml

sed -i "s/SystemdCgroup = false/SystemdCgroup = true/g" /etc/containerd/config.toml

systemctl restart containerd

ufw disable

swapoff -a

sed -i '/ swap / s/^/#/' /etc/fstab

apt-get install -y apt-transport-https curl

curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add

apt-add-repository "deb http://apt.kubernetes.io/ kubernetes-xenial main"

apt install -y kubeadm kubelet kubectl

# set the mtu size for virtual network

ip link set dev enp0s3 mtu 900

cat << EOF | tee /etc/rc.local

#! /bin/bash

swapoff -a

ip link set dev enp0s3 mtu 900

exit 0

EOF

chmod +x /etc/rc.local

**root@master:~# sh all\_kube\_nodes.sh**

**root@master:~# vim master\_kube.sh**

hostnamectl set-hostname master

kubeadm init --pod-network-cidr=192.168.0.0/16

mkdir -p $HOME/.kube

cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

chown $(id -u):$(id -g) $HOME/.kube/config

kubectl apply -f <https://docs.projectcalico.org/manifests/calico.yaml>

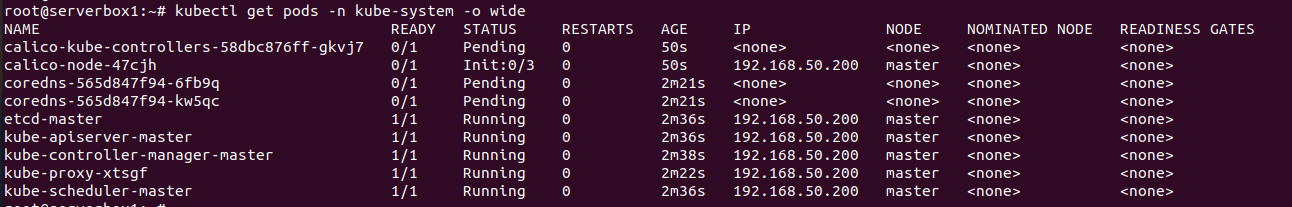
**root@master:~# sh master\_kube.sh**

**Note:**

1. Take note of the "kubeadm join xxxxx " line. We will use this line to connect workers with master.
2. Recheck cluster joining command

root@master:~# kubeadm token create --print-join-command

**root@master:~#** kubectl get pods -n kube-system -o wide



Note: This process takes lots of time. So, be patient. Wait until every pod is running.

# 2. Install softwares at all workers

**root@worker1:~#** sh all\_kube\_nodes.sh

**root@worker1:~#** hostnamectl set-hostname worker1

**root@worker1:~#** kubeadm join 192.168.50.200:6443 --token k3mz96.5uzkvqxntnv1o2li \

--discovery-token-ca-cert-hash sha256:7f8e00240837c3416699a2a0c87b981e9987effe1397e007e94d51fe90756dbf

Note: Replace this kubeadm join with your master's cluster join.

# 3. Manage PODs from Master

## Testing

**root@master:~#** kubectl get **nodes** --all-namespaces -o wide

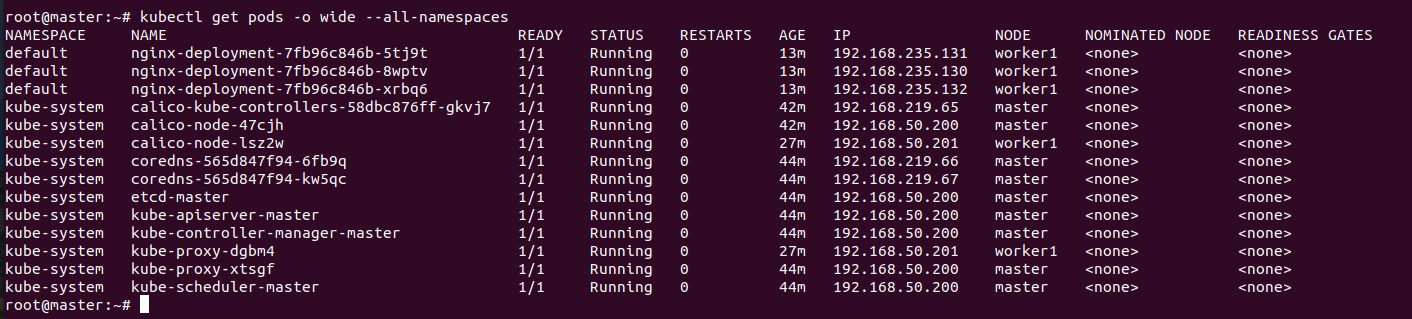
**root@master:~#** kubectl get **services** --all-namespaces -o wide

**root@master:~#** kubectl get **pods** --all-namespaces -o wide

**root@master:~#** kubectl logs nginx-deployment-7fb96c846b-24zlh

## Check POD status

**root@master:~#** kubectl get pods -n kube-system -o wide

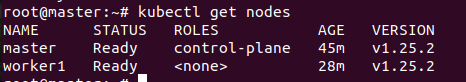


Note

Please wait a bit. It takes a long time to run all the PODs.

## Check node status

**root@master:~#** kubectl get nodes



Note

Please wait 10 minutes and check the status

# 4. Deploy pods from Master

## Run single pod

**root@master:~#** kubectl run darwin --image=nginx

**root@master:~#** kubectl get pods

**root@master:~#** kubectl get pods -o wide

# Delete a pod

**root@master:~#** kubectl delete pod darwin

## Deploy multiple pods

**root@master:~#** vim nginx-deployment.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx

labels:

app: nginx

spec:

replicas: 3

selector:

matchLabels:

app: nginx

template:

metadata:

labels:

app: nginx

spec:

containers:

- name: nginx

image: nginx:1.14.2

ports:

- containerPort: 80

# Run the deployment

kubectl create -f nginx-deployment.yaml

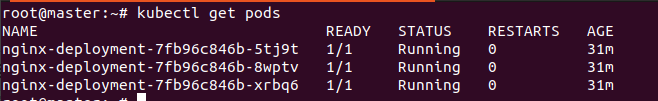
# Check the deployment status

kubectl describe deployments

# Check which pod is running on which server

kubectl get pods -o wide --all-namespaces

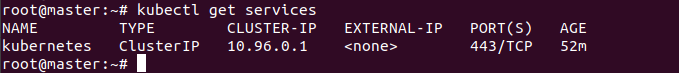
kubectl get pods



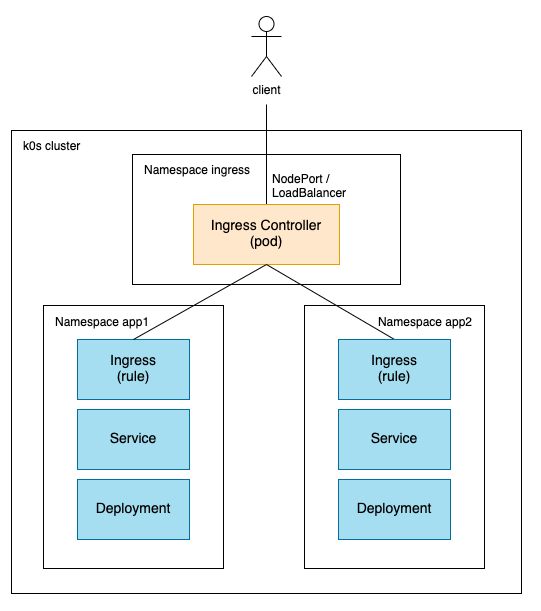
# Expose application globally

# Get the cluster ip

kubectl get services



# 5. Expose service to public/external network



1. Uncheck Enable Traefik from the Kubernetes Settings page to disable Traefik. You may need to exit and restart Rancher Desktop for the change to take effect.
2. Deploy the NGINX ingress controller via helm or kubectl.

kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.1.2/deploy/static/provider/cloud/deploy.yaml

1. Wait for the ingress pods to come up and running.

kubectl get pods --namespace=ingress-nginx

1. Create a sample deployment and the associated service.

kubectl create deployment demo --image=nginx --port=80

kubectl expose deployment demo

1. Create an ingress resource. The following command uses a host that maps to localhost.

kubectl create ingress demo-localhost --class=nginx --rule="demo.localdev.me/\*=demo:80"

1. Forward a local port to the ingress controller.

kubectl port-forward --namespace=ingress-nginx service/ingress-nginx-controller 8080:80

If you access <http://demo.localdev.me:8080/>, you should see the NGINX Welcome page.

## Install Ingress

- Install via CLI

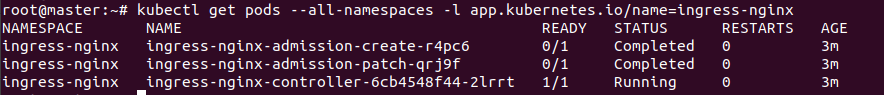
kubectl apply -f <https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.1.2/deploy/static/provider/cloud/deploy.yaml>

- [Load Balancer](https://platform9.com/learn/v1.0/tutorials/nginix-controller-via-yaml#load-balancer)

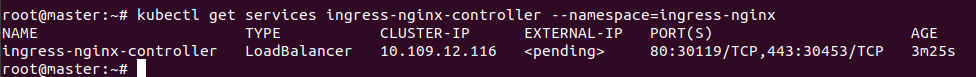
kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.1.2/deploy/static/provider/cloud/deploy.yaml

- Check the PODs status

kubectl get pods --all-namespaces -l app.kubernetes.io/name=ingress-nginx



kubectl get services ingress-nginx-controller --namespace=ingress-nginx



## Deploy PODs

root@master:~# vim my\_deployment.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

name: hello-world

labels:

app: hello-world

spec:

replicas: 2

selector:

matchLabels:

app: hello-world

template:

metadata:

labels:

app: hello-world

spec:

containers:

- name: hello-world

image: gcr.io/google-samples/node-hello:1.0

ports:

- containerPort: 8080

root@master:~# kubectl apply -f my\_deployment.yaml

## Connect service with PODs

root@master:~# vim my\_service.yaml

apiVersion: v1

kind: Service

metadata:

name: hello-world

spec:

ports:

- port: 80

targetPort: 8080

selector:

app: hello-world

root@master:~# kubectl apply -f my\_service.yaml

## Connect ingress with service

### http

root@master:~# vim my\_http.yaml

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: hello-world

annotations:

spec:

ingressClassName: nginx

rules:

- host: host1.domain.ext

http:

paths:

- pathType: Prefix

path: "/"

backend:

service:

name: hello-world

port:

number: 80

root@master:~# kubectl apply -f my\_http.yaml

## https

root@master:~# vim my\_https.yaml

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: hello-world

annotations:

cert-manager.io/cluster-issuer: letsencrypt-prod

spec:

ingressClassName: nginx

tls:

- hosts:

- host1.domain.ext

- host2.domain.ext

secretName: hello-kubernetes-tls

rules:

- host: host1.domain.ext

http:

paths:

- pathType: Prefix

path: "/"

backend:

service:

name: hello-world

port:

number: 80

root@master:~# kubectl apply -f my\_https.yaml

# 6. Testing

root@master:~# kubectl get **nodes** --all-namespaces -o wide

root@master:~# kubectl get **services** --all-namespaces -o wide

root@master:~# kubectl get **pods** --all-namespaces -o wide

root@master:~# kubectl logs nginx-deployment-7fb96c846b-24zlh

# 7. Reference

**Ingress basic**

https://platform9.com/blog/building-a-complete-stack-ingress-controllers/

**Kubernetes networking**

https://www.youtube.com/watch?v=NPFbYpb0I7w